Edinburgh Napier University

SET08101 Web Tech

Lab 1 - Introduction & Learning Environment

Dr Simon Wells

1 Aims

At the end of the practical portion of this topic you will:

- Be able to log into the University system
- Be able to run simple programs from the command line
- Remember how to program simple software
- Have retrieved some supplementary texts from an online publisher

NOTICE: This is an introductory lab that is designed to ease ourselves back into programming and make sure that we are comfortable with our learning environment before we start learning new things in earnest after the Xmas break. Things will get harder and more complex over the coming weeks but we need to ensure that we have some tools in place to start with.

2 Activities

2.1 Get some tools

All we really need for front-end development is a decent text editor and a web browser. Obviously as things get more complicated we will extend our toolbox, but this is a good place to start. If you don't already have a favourite editor (and you are on Windows) then Notepad++[1] is a goodplace to start. Download it, install it. You can even run it from a USB stick so that you can take your development environment around with you.

2.2 Hello Web

Write a short HTML document called hello.html and save it in a folder called lab1, e.g.

```
1 <html>
2     <head>
3         <title>WEB TECH</title>
4     <head>
5     <body>
6         <h1>Hello Web Tech</h1>
7     </body>
8 <html>
```

Open your web page in a variety of browsers (whichever you have installed on the machine you are using). Have a look to see what browsers are installed on the lab machines and compare any differences in the page that is rendered. Consder why different browsers might display things differently.

2.3 Developer Tools

You will be using the developer tools built into Chrome a lot in this module, often to find out why things aren't running properly (or why the browser isn't doing what you're sure you told it to do). Partly this is because you will make mistakes. Partly because getting lots of different web technologies to work together is tricky. Things don't fit together just right and we need to work out why. We'll use Chrome to do this because:

- It runs on nearly every common platform
- It has developer tools built in
- It's a fairly stable platform to work with

[1] https://notepad-plus-plus.org/

Open your hello.html file in Chrome and view it using the developer tools. Try mangling your source file, e.g. removing some tags and compare your input file to the view in the developer tools. The browser seems to fix issues, why do you think this is? How mangled can your file be before it isn't displayed?

2.4 Background Reading

Use your browser to visit `http://link.springer.com`. If you are on the university network you shouldn't have to log in, but otherwise your university credentials will work with the Shibboleth log in option on the site. Use the search option with the following query: "Algorithms and Data Structures", you might want to narrow the result by selecting the "Books" option in the menu on the lefthand side.

You can, and should, download PDF (or EPUB) copies of the following books which will supplement your reading throughout the module.

- Introducing Web Development
- Practical Web Design for Absolute Beginners
- Moving to Responsive Web Design
- Beginning Responsive Web Design with HTML 5 and CSS3
- Sustainable Web Ecosystem Design

Pointers to other texts, chapters, and papers will be suggested throughout the module, but these should be enough to get started and provide a solid underpinning for your learning.

2.5 Use Git

Git is a source control system that enables you to keep track of your source code, its history and any changes you make. Git can be used to track any file but is most efficient and best suited when used only with textual files. Because Git is a *distributed* source control system it works very well to enable groups of people to work on the same source code as well as supporting experimenting with your code, trying out lots of different ideas in separate *branches* (which are a bit like a copy of your code but with tools to help manage that copy and support re-integrating it with your main source tree if you want to), and being able to roll back to an earlier version if you decide you have take a wrong turn.

I'm not going to give detailed instrutions for setting up Git on the lab machines, that's why this is a bit of a *challenge*.

> **IMPORTANT** Git will be a requirement for submission of the coursework hand-in so should spend some time getting familiar with it as soon as possible. A good place to start is by dipping into the Git SCM book[2].

There are also numerous interactive tutorials and resources to help you get started with Git:

1. Github's Learn Git 15 minute tutorial: `https://try.github.io/levels/1/challenges/1`
2. Learn Git Branching `http://pcottle.github.io/learnGitBranching/`
3. Git Immersion `http://gitimmersion.com/lab_01.html`

You should also create either a Github account[3] or a Bitbucket account[4] (or both if you like) then create a repository within your new account called 'set08101'. You can then push all of your code throughout the module into this repository. I'd suggest adding a folder for todays lab and commiting your hello.html file.

[3] `https://github.com/`
[4] `https://bitbucket.org`

Using Git counts as practising a professional skill. The advantage of this approach is that at any point, if you need help with your code, then you have a copy that can be shared remotely, e.g. with Simon or another member of the teaching team. However this only works if you keep adding your code to your repository. That means whenever you make changes you need to (1) add them, (2) commit them with an explanatory message, and (3) push the changes from your local repository to the shared one on Github or Bitbucket.